Please replace the paragraph beginning at line 13 of page 56, with the following rewritten paragraph:

*A22*

--Derived from the HcsLogPort are the HcsResourceLogPort and HcsLogFacilityPort classes. The HcsResourceLogPort provides easy support for the resource developer while the HcsLogFacilityPort provides direct access to HcsLogFacilities. In general, the first is used by Resource derivations while the second would be used by resource servers and system components.--

Please replace the paragraph beginning at line 18 of page 59, with the following rewritten paragraph:

*A23*

--HcsResourceImp::putLogRecord implementation: keeps an internal HcsLogFacilityPort instance which is used to forward records through. Before a record is actually forwarded, HcsResourceImp checks to make sure an HcsLogFacility has been assigned to its port. If this is not the case, an attempt is made to locate the local HcsLogFacility for that node via the HcsResrcBusIf::locateLocalResourceById method (new). In any case, the log record is written via the resource instance's HcsLogFacilityPort.--

**IN THE CLAIMS:**

Please amend claims 1 to 7 as follows:

1.    (Amended) A system for managing software components in a distributed computing environment, the system comprising:

a tracking system for tracking when a software component changes state and for providing a state change notification of a change in state of the tracked software component; and

a property notification system for providing a property notification to the software component when a property of another software component is set wherein software components of the system use services of both the tracking system and the property notification system.

2.    (Amended) The system of claim 1, further including:

an event notification system for providing an event notification to the software component when at least one of the software component and another software component generates an event.

-12-

3.      (Amended) The system of claim 1, further including:

a logging system for logging records of activity of the software component.


4.      (Amended) The system of claim 1, further including:

a directory component that receives a tracking reference and returns a corresponding

behavioral reference.


5.      (Amended) A system for managing software components in a distributed computing

environment, the system comprising:

a tracking system for tracking when a software component changes state and for

providing a state change notification of a change in state of the tracked software component; and

an event notification system for providing an event notification to the software

component when another software component generates an event, wherein software components

of the system use the services of the tracking system and the event notification system.


6.      (Amended) The system of claim 5, further including:

a logging system for logging records of activity of the software component.


7.      (Amended) The system of claim 5, further including:

a directory component that receives a tracking reference and returns a corresponding

behavioral reference.


Please add new claims 8 to 39, as follows:

--8.      (New) In a system for a distributed computing environment, wherein the system includes

a communications bus, a bus manager having at least one bus management component, at least

one server node and at least one client node, wherein said at least one server node, said at least

one client node and said at least one bus management component are interconnected via said

-13-

communications bus, and wherein each of said at least one client node includes at least one client resource for requesting notification when at least one of an event is generated, a server resource of said at least one server node changes state or a server resource of said at least one server node changes a property, a method for managing resources of the system, comprising:

tracking when a software component changes state and providing a state change notification of a change in state of the tracked software component; and

providing a property notification to the software component when a property of at least one of the software component and another software component is set.

9.    (New) A method according to claim 8, further comprising:

providing an event notification to the software component when at least one of the software component and another software component generates an event.

10.    (New) A method according to claim 8, wherein said tracking includes:

receiving by the system a request from the client to track a state of the object;

watching the state of the object to detect when the object enters the up state and when the object enters the up state, first performing at least one behavior that is specified by the client to be performed when the object enters the up state and when the object is in the up state, monitoring the state of the object by the system to detect when the object enters the down state; and

monitoring the state of the object to detect when the object enters the down state, and when the object enters the down state, second performing at least one behavior that is specified by the client to be performed when the object enters the down state.

11.    (New) A method according to claim 8, wherein said providing a property notification includes:

first registering by the client resource to track a server resource;

after the server resource enters the up state, second registering by the client resource to

watch a property of the server resource; and

after the property of the server resource is set, invoking by the server resource a property set function of the client resource.

12.     (New) A method according to claim 9, wherein said providing an event notification includes:

registering by a client resource an interest in an event type;

upon the occurrence of an event classified with said event type, providing an asynchronous event signal that is distributed to all client resources that have registered to listen for the signal.

13.     (New) A method according to claim 12, wherein each event has an associated event type whereby event types are aggregated types.

14.     (New) A method according to claim 13, wherein each event has an associated event type whereby event types are hierarchically organized.

15.     (New) A method according to claim 14, wherein registering by a client resource for a particular event type includes registering the client resource for all event types falling with the hierarchical classification for the particular event type.

16.     (New) A method according to claim 14, wherein event types include a timer event type.

17.     (New) A method according to claim 16, wherein a timer event type is one of a catastrophic timer event, a warning timer event and an informational timer event.

18.     (New) A method according to claim 17, wherein an informational timer event is one of a start-up timer event and a shut-down timer event.

19.    (New) A method according to claim 14, wherein the hierarchy of an event type embedded in the identification of the event type.

20.    (New) A method according to claim 9, wherein said providing an event notification includes:

registering by a client resource of a client node with a listener component an interest in listening for an event, wherein said registering includes invoking a listen message and specifying an event type to the listener component; and

receiving by the client resource from the listener component an event notify message along with event information when an event of that event type is generated.

21.    (New) A method according to claim 20, wherein said providing an event notification further includes:

un-registering by the client resource the interest in listening for the event, wherein said un-registering includes invoking a stop listening message along and specifying the event type to the listener component.

22.    (New) A method according to claim 21, wherein each client node has a listener component through which is routed all event related messages for all client resources registered to listen for events on the node.

23.    (New) A method according to claim 22, wherein a listener component of a client node routes event-related messages to a listener bus management component whereby the listener component notifies the listener bus management component to listen for all event types for which client resources are registered to listen for events on the client node.

24.    (New) A method according to claim 23, wherein each listener component includes a

listener table cache that contains a mapping from each event type for which a listen request has been registered and for each client that has registered for the event type, such that when the listener component receives an event notification, the listener component accesses the listener table cache to notify each client resource registered to listen for the event type.

25. (New) A method according to claim 23, wherein the listener bus management component includes a listener table that contains a mapping from each event type to the registering client nodes such that when the listener bus management component receives an event posting, the listener bus management component notifies each node that has registered to listen for events of the event type and for events of any event type that is aggregated within the event type.

26. (New) A method according to claim 25, wherein the listener bus management component notifies each node that has registered to listen for events of the event type and for events of any event type that is a hierarchical parent of the event type.

27. (New) A method according to claim 8, further including:
    logging activity of the software component in at least one log record.

28. (New) A method according to claim 27, wherein each of the at least one log record comprises at least one of: type information, time information, creator information and text information.

29. (New) A method according to claim 27, wherein said logging includes logging at a local log facility and logging at a central log facility.

30. (New) A method according to claim 29, wherein each client node includes an instance of a local log facility which receives all log records from various software components on the client node buffers the log records until they are transmitted to the central log facility.

31.     (New) A method according to claim 30, wherein a local log facility is configured to forward its records to another local log facility instead of the central log facility.

32.     (New) A method according to claim 29, wherein there is only one instance of the central log facility for the whole system which accepts all log records from all components within the system and whereby the central log facility provides for the final storage of logging information.

33.     (New) A method according to claim 29, wherein the central log facility provides long term, off line, archival of the entire system.

34.     (New) A method according to claim 29, wherein the central log facility provide for on-line viewing of a desired portion of the log records.

35.     (New) A method according to claim 27, wherein said logging is capable of being disabled.

36.     (New) A method according to claim 8, wherein a server node of said at least one server node is also a client node of said at least one client node.

37.     (New) A computer readable medium comprising computer executable instructions for performing the method of claim 8.  .

38.     (New) A modulated data signal carrying computer executable instructions for performing the method of claim 8.

39.     (New) A computing device comprising means for performing the method of claim 8.--